```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <math.h>
#include "graph.h"


Graph::Graph( void ) {  //make a random test pattern;
      int i,j;
      srand( (unsigned int) time( NULL ) );
      N = (int) (rand()%6)+3; // the maximum number of nodes : 8
      connected = (char **) malloc( N*sizeof(char *) );
      for ( i = 0; i < N; i++)
            connected[i] = (char *) malloc( N*sizeof(char) );

      for ( i = 0; i < N; i++) { // the graph is randomly generated
            connected[i][i] = 1;
            for ( j = i+1; j < N; j++ ) {
                  if ( rand()%2 == 1 ) {
                        connected[i][j] = 1;
                        connected[j][i] = 1;
                  }
                  else {
                        connected[i][j] = 0;
                        connected[j][i] = 0;
                  }
            }
      }
      print();
      compsub.nodeInit( N );
}

Graph::Graph( char* fileName ) { // make a graph from a file
      char c[1000];
      int i = 0;
      int row = 0;
      int col,j;
      FILE* fp;
      N = 0;
      if ( ( fp = fopen(fileName, "rt") ) == NULL ) {
            printf( "FileOpenError!\n" );
            exit(-1);
      }
      while (!feof(fp)) {
            fgets( c, 1000, fp );
            if ( N == 0 ) {
                  i = 0;
                  while( c[i] != '\n' ) {
                        if ( c[i]=='0' || c[i]=='1' )
                              N++;
                        i++;
                  }
                  if ( N == 0 ) {
                        printf( "TheFileIsCorrupted!\n");
                        exit(-1);
                  }
                  connected = (char **) malloc( N*sizeof(char *) );
            }
            if ( row < N )
                  connected[row] = (char *) malloc( N*sizeof(char) );
            col = 0;
```

```
            i = 0;
            while( c[i] != '\n' ) {
                  if ( c[i]=='0') {
                        if ( col>=N || row >= N) {
                              printf( "TheFileIsCorrupted!\n");
                              exit(-1);
                        }
                        connected[row][col++] = 0;
                  }
                  if ( c[i]=='1') {
                        if ( col>=N || row >= N) {
                              printf( "TheFileIsCorrupted!\n");
                              exit(-1);
                        }
                        connected[row][col++] = 1;
                  }
                  i++;
            }
            row++;
      }
      fclose( fp );
      print();
      compsub.nodeInit( N );
}

Graph::~Graph( void ) {
      int i;
      for ( i = 0; i < N; i++ )
            free( connected[i] );
      free( connected );
}

char Graph::areConnected( int nodeA, int nodeB ) {
      if ( nodeA >= N || nodeB >= N ) {
            printf( "AccessViolationInNodeConnectionCheck\n");
            exit(-1);
      }
      return connected[nodeA][nodeB];
}

int Graph::size( void ) {
      return N;
}

void Graph::print( void ) {
      int i,j;
      printf("\n");
      for ( i = 0; i < N; i++ ) {
            for ( j = 0; j < N; j++ ) {
                  if (connected[i][j])
                        printf("1");
                  else printf("0");
            }
            printf("\n");
      }
      printf("\n");
}
```